# Lec 1 Compilers

→ what is Compilers?

→ Phases of Compilers?

→ function of phases

→ our goals.

→ The Programs we use to process it :)

Translators → do the same function of Compilers.

→ Types of Translators? their functions?

Preprocessor :-

micro inst ياخد ← ويحولها بتاعتها.

in Programming # include .... السطر ده بتاع (Preprocessor)

← ( translation) بعمل يقدر

(another high level language) لل ( high level language)

→ some Compilers can give us executable code (ready) or assembly code (need some operations to be ready)

→ Translator has to get two programs do the same function to translate between them.

## Types of Translator

→ Pre Processor

(level language) to (level language) يحول من ←

→ Assembler

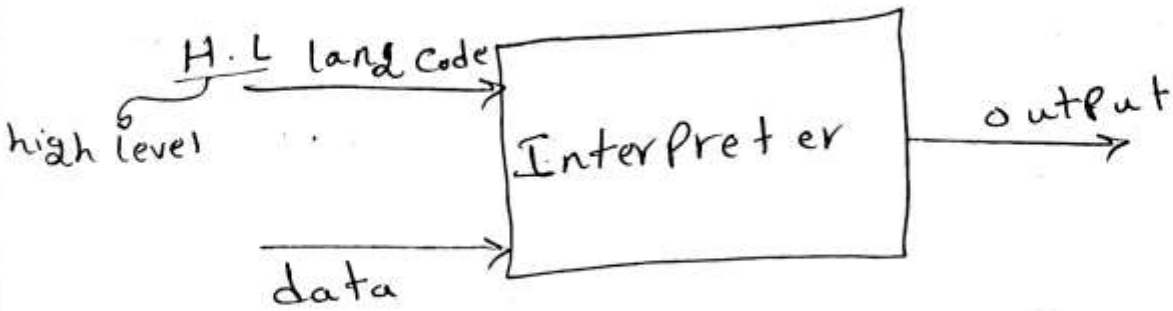
→ It is one to one

(one instruction) يقابل (assembly inst.) كل

(assembler) يقوم بعكس ← (Disassembler)

(Assembler) ← مش معنى انه يطلع (o/p) احنا نفذه

(machine) على (run) ونطلع o/p.




→ difference between Compiler & interpreter

↳ on pdf slides.

(run) أسرع و أول ( interpreter) ←

بس ← (in total) الـ ( Compiler ) أسرع .

← لو الـ (interpreter) قابل (error) يؤثر على خرجه يتوقف تماماً أما الـ (Compiler) فيكمل العمل.

Compilers

→ classified for instruction

a) single Pass.

b) multiple Pass.

→ classified with Considration of optimization:-

a) Global b) local

→ There are anothers do the same function or similar function to the compilers.

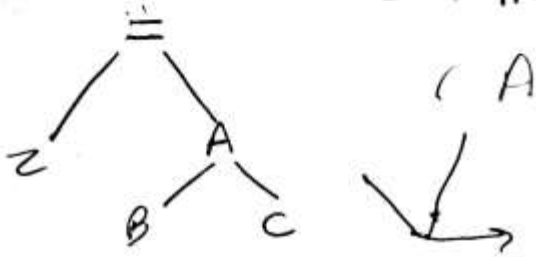
Like cross-compiler, source-to-source.

→ Preprocessor is similar to source-to-source in the definition.

→ difference between source-to-source compiler and preprocessor ⇒ report.

## Architecture of compilers

→ front-end → away from machine
→ machine independent.




( ADD (A, B, C)

→ intermediate representation
→ not related to machine.

→ Analysis go in three parts.

Analysies → lexical (scanner)
→ syntax
→ semantic

Front end → Analysis
→ intermediate code representation.

← القياس ما بين (front end) و (back-end)
هو علاقته بال (machine).

back-end → if we work with local optimization.

→ what is Parsing ??

→ syntax-analysis ⇒ it's 2nd name is Parsing.

→ machine independent → global optimiser.

→ machine dependent → local "

⇒ Phases we will study more:-

* lexical
* syntax
* Code Generator
* machine dependent Code optimiser.

## → Compiler Function

→ transform from high level lang. to low level lang.

→ handling for error.

→ build symbol table.

## * Symbol table

← أي (variable) ← يخزنه في (symbole table) ← يحجز مكان في الذاكرة ثم يتم تخزين الـ (Symbole table) في الذاكرة أي أنه يأخذ مكان في الـ (memory).

→ search for it.

→ Lexical (scanner)

الـ tokens هنا يعني الـ (word) لكن في البرنامج فقط.

→ takes statement by statement.

تتشوف الـ (Lexical analysis) فاضي (command) معناه إنه مش بيشيله. (Preprocessor)

→ lexemes must be meaning ful as a language.

→ syntax analysis

→ check source code for errors.

→ give us parsal tree for atoms as o/p.

→ search of the sentence's structure.

intermediate lang → machine independent.

← هذه الخلاصة عبارة عن ملاحظات تساعد على فهم ال pdf فقط.

7 Lec 1